Fast Consensus under Eventually Stabilizing Message Adversaries

Manfred Schwarz
TU Wien
Vienna, Austria
mschwarz@ecs.tuwien.ac.at

Kyrill Winkler
TU Wien
Vienna, Austria
kwinkler@ecs.tuwien.ac.at

Ulrich Schmid
TU Wien
Vienna, Austria
s@ecs.tuwien.ac.at

ABSTRACT

This paper is devoted to deterministic consensus in synchronous dynamic networks with unidirectional links, which are under the control of an omniscient message adversary. Motivated by unpredictable node/system initialization times and long-lasting periods of massive transient faults, we consider message adversaries that guarantee periods of less erratic message loss only *eventually*: We present a tight bound of $2D+1$ for the termination time of consensus under a message adversary that eventually guarantees a single vertex-stable root component with dynamic network diameter D, as well as a simple algorithm that matches this bound. It effectively halves the termination time $4D+1$ achieved by an existing consensus algorithm, which also works under our message adversary. We also introduce a generalized, considerably stronger variant of our message adversary, and show that our new algorithm, unlike the existing one, still works correctly under it.

1 Introduction

We study deterministic distributed consensus in synchronous dynamic networks connected by unreliable, *unidirectional* links. Assuming unidirectional communication, in contrast to most existing research [10, 12], is not only of theoretical interest: According to [16], 80% of the links in a typical wireless network are sometimes asymmetric. In fact, in wireless settings with low node density, various interferers and obstacles that severely inhibit communication, as in disaster relief applications [15], for example, bidirectional links may simply not be achievable. Moreover, implementing low-level bidirectional communication between every pair of nodes is costly in terms of energy consumption, delay time and hardware resources. It may hence be an overkill for applications that just need some piece of information available at one node to reach some other node, as this is also achievable via directed multi-hop paths. Obviously, in such settings, algorithmic solutions that do not assume bidirectional single-hop communication in the first place provide significant advantages.

In this paper, we model directed dynamic networks as synchronous distributed systems made up of n processes, where processes have no knowledge of n. In every round, the processes attempt a full message exchange and compute a new local state based on the messages successfully received in the message exchange. The actual communication in round $r = 1, 2, \ldots$ is modeled as a sequence of directed communication graphs $\mathcal{G}^1, \mathcal{G}^2, \ldots$, which are considered under the control of a omniscient *message adversary* [1, 17]: The message adversary determines which messages are delivered and which get lost in each round.

In contrast to [1], where message adversaries are oblivious in the sense that they can choose the round graphs arbitrarily from a *fixed* set of candidates only, this paper, inspired by the research in [3, 4], considers message adversaries that may pick the graphs generated in some round depending on the particular round number. Obviously, this allows to model *stabilizing behavior*, which is not only of theoretical interest but also relevant from a practical point of view: Starting-up a real dynamic distributed system is likely a quite chaotic process, as nodes boot at different times and execute various initialization procedures. One can expect, though, that the system will operate in a better orchestrated way after some unpredictable startup time. A similar effect can be expected after a period of excessive transient faults, as caused by the abundant ionizing particles emitted during heavy solar flares [2, 8], for example. In this paper, we hence focus on stabilizing message adversaries, which allow finite initial periods where arbitrary graphs may be generated.

The distributed computing problem considered in this paper is consensus. A consensus algorithm ensures that all processes in the system eventually agree on a common decision value, which is computed (deterministically) from local inputs. It is an important primitive for any distributed application where data consistency is crucial. Unlike in dynamic networks with unreliable *bidirectional* links, where solving consensus is relatively easy [12], solving consensus under message adversaries that generate unreliable directed links is inherently difficult: For example, it is impossible to solve synchronous deterministic consensus with two processes connected by a pair of lossy directional links [18], even when it is guaranteed that only one link can fail in every round [19]. Therefore, in order to solve consensus, the power of the adversary must be restricted somehow. Exploring the solvability/impossibility-border for consensus in directed dynamic networks is hence an interesting and challenging topic.

Contributions

(1) We present two variants of a "natural" stabilizing message adversary, which takes into consideration the eventually stabilizing behavior that can reasonably be expected from real dynamic networks. During some finite initial period, the communication graphs can be (almost) arbitrary: In particular, they may contain any number of *root com-*

ponents[1] (strongly connected components that have no incoming edges from outside of the component), which may even consist of the same set of nodes (with possibly varying interconnect topology) for up to D consecutive rounds. $1 \leqslant D < n$ is a system parameter, known to the processes, which ensures that information from all members of a single root component that remains the same for at least D rounds reaches all n processes in the system. The "chaotic" initial period ends, at some unknown stabilization round r_{sr}, when, for the first time, a single root component R occurs that consists of the same set of processes for more than D consecutive rounds.

The simple *eventually stable forever after* variant of our message adversary, $\Diamond\texttt{STABLE}(D)$, guarantees that R remains a root component in all rounds after r_{sr}. $\Diamond\texttt{STABLE}(D)$ is quite restricted in its behavior after stabilization, but is easy to analyze and facilitates an easy comparison of the performance (in particular, of the termination times) of different consensus algorithms. The rigid properties of $\Diamond\texttt{STABLE}(D)$ are relaxed considerably in the case of our message adversary $\Diamond\texttt{STABLE}'(D)$, which just requires that R re-appears, as a single root component, in at least D (arbitrary, i.e., non-consecutive) rounds in the execution suffix after $r_{\mathrm{sr}} + D$.

(2) We prove that no consensus algorithm can terminate under $\Diamond\texttt{STABLE}(D)$ (and hence under $\Diamond\texttt{STABLE}'(D)$) before $r_{\mathrm{sr}} + 2D$. Note that the fastest known algorithm to date was presented in [4] and also works under $\Diamond\texttt{STABLE}(D)$. It has a termination time of $r_{\mathrm{sr}} + 4D$ and is hence sub-optimal here.

(3) We provide a simple consensus algorithm, which matches the termination time lower bound of $2D+1$ under $\Diamond\texttt{STABLE}(D)$ and works correctly also under $\Diamond\texttt{STABLE}'(D)$. Note that the algorithm from [4] fails under $\Diamond\texttt{STABLE}'(D)$, even though its code is considerably more complex.

Previous results

In [3], Biely et.al. showed that consensus is solvable under a message adversary that generates graphs containing a single root component only, which eventually consists of the same processes for at least $4D$ consecutive rounds; the term *$4D$-vertex-stable root component* has been coined to reflect this fact. Note that vertex-stable root components neither imply a static network nor a stable subgraph over multiple rounds. It has also been shown in [3] that consensus is impossible if the adversary is not forced to generate a root component that is vertex-stable for at least D rounds.

In [4], we showed that consensus can be solved under a message adversary that may generate multiple vertex-stable roots, albeit with a worse worst case termination time and a far more complex algorithm. More specifically, the message adversary proposed in this paper guarantees root components that (i) are eventually stable for at least $4D$ rounds concurrently, and (ii) ensures some distinct information flow between successive vertex-stable root components ("majority influence"). The proposed algorithm is gracefully degrading, in the sense that it solves k-set agreement for the worst-case optimal choice of k, when consensus ($k = 1$) cannot be solved in the given run. Recall that in k-set agreement, the consensus agreement condition is relaxed such that up to k different decision values are permitted.

Other related work

Dynamic networks have been studied intensively in distributed computing (see the overview by Kuhn and Oshman [11] and the references therein). Besides work on peer-to-peer networks like [13], where the dynamicity of nodes (churn) is the primary concern, different approaches for modeling dynamic connectivity have been proposed, both in the networking context and in the context of classic distributed computing. T-interval-connectivity in synchronous distributed computations has been introduced in [10].

Agreement problems in dynamic networks with undirected communication graphs have been studied in the work by Kuhn et al. [12]; it focuses on the Δ-coordinated consensus problem, which extends consensus by requiring all processes to decide within Δ rounds of the first decision. Agreement in directed graphs has been considered in [1,3,4,6,17,19,20]. Whereas [6, 19] considerably restrict the dynamicity of the communication graphs, e.g., by not allowing stabilizing behavior, which effectively causes them to belong to quite strong classes of network assumptions in the classification of Casteigts et al. [5], the algorithms of [3, 4, 20] allow to solve consensus under very weak network assumptions: [3] only admits single-rooted graphs, whereas [4] provides a consensus algorithm that gracefully degrades to k-set agreement in unfavorable runs under a fairly strong stabilizing message adversary. Afek and Gafni [1] introduced (oblivious) message adversaries for specifying network assumptions in this context, and used them for relating problems solvable in wait-free read-write shared memory systems to those solvable in message-passing systems. Raynal and Stainer [17] used message adversaries for exploring the relationship between round-based models and failure detectors.

2 Model

We model a synchronous message passing system as a set Π of $|\Pi| = n > 1$ deterministic state machines, called *processes*. Processes do not necessarily know n but have unique identifiers that we pick, w.l.o.g., from the set $\{1, \ldots, n\}$. In our analysis, we use a process and its identifier interchangeably when there is no ambiguity. Processes operate in lock-step rounds, where each round consists of a phase of full message exchange, followed by an instantaneous local *computing step*. Following [3, 4], the actual communication in round $r \geqslant 1$ is according to a digraph[2] $\mathcal{G}^r = (V, E^r)$ controlled by an omniscient *message adversary*: Each vertex in V corresponds to exactly one process of Π, and an edge from p to q, denoted $(p \to q)$, is present in E^r iff the adversary permits the delivery of the message sent from p to q in round r. We assume that $\mathcal{G}^r$ contains self-loops $(p \to p)$ for all $p \in V$, i.e., processes always receive their own message in every round. Rounds are communication-closed, i.e., messages sent in some round r and delivered in a later round $r' > r$ are dropped.

The messages sent and the state transitions performed by the processes in a round are guided by a deterministic message-sending and state-transition function, respectively, which are specified implicitly by algorithms in pseudo-code:

[1] Note that root components have already been used in the asynchronous consensus algorithm for a minority of initially dead processes introduced by Fischer, Lynch and Paterson in [9].

[2] Usually, we sloppily write $p \in \mathcal{G}^r$, resp. $(p \to q) \in \mathcal{G}^r$ instead of $p \in V$ resp. $(p \to q) \in E^r$.

The *local state* of a process comprises all its local variables; the *message-sending function* determines the message to be broadcast in a round, and the *state-transition function* determines the local state reached at the end of the round, depending on the previous state and the set of messages received in the round. Most of the time, we will assume that the algorithms are *full-information*, i.e., processes keep track of received messages and forward their entire states to all processes they can reach in every round.

In our analysis, p^r denotes the *local state* of process p at the end of round $r \geqslant 1$, after its computing step; p^0 is the initial state at the beginning of round 1. The value of a particular variable *var* in p^r is denoted by var_p^r.[3] The vector of states of all the processes at the end of round r is called round r *configuration* C^r; C^0 denotes the initial configuration. An *execution*, or *run*, is an alternating sequence of configurations and communication graphs. As our algorithms are deterministic, it is uniquely determined by a given initial configuration C^0 together with an infinite sequence[4] of communication graphs $(\mathcal{G}^r)_{r=1}^{\infty}$, which is controlled by a *message adversary*. More generally, any execution segment, starting from configuration C^r, is uniquely specified by a tuple like $\langle C^r, (\mathcal{G}^i)_{i=r+1}^{a}, (\mathcal{G}^j)_{j=a+1}^{b}, \ldots\rangle$. An execution is called *admissible*, if it is in accordance with the message-sending and state-transition functions of the processes and the definition of the message adversary.

As in [4], we will restrict the power of a message adversary in terms of the properties of the sequences of communication graphs it may legitimately generate. Consequently, an adversary A that has a set of properties P_A can formally be specified via the set of its *feasible* infinite communication graph sequences $A := \{(\mathcal{G}^r)_{r=1}^{\infty} \mid (\mathcal{G}^r)_{r=1}^{\infty} \text{ satisfies } P_A\}$. We say that an adversary A is weaker than an adversary B, resp. that B is stronger than A, if all feasible sequences of A are also in B but not vice-versa, i.e., $A \subset B$. If A contains sequences not in B and B contains sequences not in A, A and B are incomparable. An example for two incomparable adversaries is the adversary that allows only chains for each $\mathcal{G}^r$ and the adversary that allows only circles for each $\mathcal{G}^r$.

We say that a problem is *impossible* under some message adversary if there is no deteministic algorithm that solves the problem for every feasible communication graph sequence. For example, every problem that requires at least some communication among the processes is impossible under the unrestricted message adversary, which may generate all possible graph sequences: The sequence $(\mathcal{G}^r)_{r=1}^{\infty}$ where no $\mathcal{G}^r$ contains even a single edge is also feasible here.

We are interested in solving the *consensus problem*, where each process p has an initial value x_p and a write-once decision value y_p in its local state. Formally, the following conditions must be met in every execution of a correct consensus algorithm in our setting for $p, q \in \Pi$:

(Agreement) If p assigns value v_p to y_p and q assigns v_q to y_q, then $v_p = v_q$.

(Termination) Eventually, every p assigns a value to y_p.

(Validity) If p assigns a value v to y_p, then there is some q such that $x_q = v$.

[3]Note that, throughout our paper, superscripts usually denote round numbers, with the implicit assumption that they refer to the end of a round (after the computing step), whereas subscripts typically identify processes.

[4]As usual, we denote by $(\mathcal{G}^r)_{r=a}^{b}$ the sequence $(\mathcal{G}^a, \ldots, \mathcal{G}^b)$ of communication graphs.

Dynamic graph concepts

As in [3,4], the message adversaries considered in this paper will focus on *root components* in the communication graphs, which are strongly connected components that have no incoming edges. Their importance has already been recognized in the celebrated paper [9] by Fischer, Lynch and Paterson, which also introduces an algorithm for asynchronous consensus with a minority of initially dead processes. It essentially identifies the (unique) root component in the initial communication graph formed by the processes waiting for first $n/2$ messages to arrive.

Definition 1 (Root Component). *A non-empty set of nodes $R \subseteq V$ is called a round r root component of $\mathcal{G}^r$, if it is the set of vertices of a strongly connected component $\mathcal{R}$ of $\mathcal{G}^r$ and $\forall p \in \mathcal{G}^r, q \in R : (p \to q) \in \mathcal{G}^r \Rightarrow p \in R$. We denote by $\mathrm{roots}(\mathcal{G}^r)$ the set of all root components of $\mathcal{G}^r$, resp. the single root component of $\mathcal{G}^r$, and by $|R|$ the number of nodes in R.*

By contracting the strongly connected components of $\mathcal{G}^r$, it is easy to see that every graph has at least one root component (just called "roots" for brevity). Furthermore, if $\mathcal{G}^r$ contains a single root only, contraction leads to a tree, so $\mathcal{G}^r$ must be weakly connected in this case.

Corollary 1. *For any directed graph $\mathcal{G}^r$, $|\mathrm{roots}(\mathcal{G}^r)| \geqslant 1$, and if $|\mathrm{roots}(\mathcal{G}^r)| = 1$, then $\mathcal{G}^r$ is weakly connected.*

We call a set of nodes R that forms a root component in every communication graph of a sequence $(\mathcal{G}^r)_{r\in I}$ a *common root* of this sequence. Note carefully that the interconnect topology of the nodes in R, i.e., the root component $\mathcal{R}$ taken as a subgraph of $\mathcal{G}^r$, as well as the outgoing edges to the remaining nodes $\Pi \setminus R$ in $\mathcal{G}^r$, may be different in every round r in the sequence. The index set I of rounds in $(\mathcal{G}^r)_{r\in I}$ is usually an interval $I = [a, b]$ of $|I| = b - a + 1$ consecutive rounds[5] (we will call $(\mathcal{G}^r)_{r\in I}$ a *consecutive* graph sequence in this case), but can also be an arbitrary index set that is ordered according to increasing round numbers. If a consecutive graph sequence is maximal wrt. R being its common root, we call R a maximal common root.

Definition 2 (Common root). *We say that a sequence $(\mathcal{G}^r)_{r\in I}$ has a common root R, iff there exists a root R (with possibly different interconnect topology) such that $R \in \mathrm{roots}(\mathcal{G}^r)$ for all $r \in I$. If $I = [a, b]$ with $|I| = b - a + 1$ is an interval of consecutive rounds $a, a + 1, \ldots, b$, $(\mathcal{G}^r)_{r\in I}$ is called a consecutive graph sequence. We call R a maximal common root of a consecutive graph sequence $(\mathcal{G}^r)_{r=a}^{b}$, iff R is a common root of $(\mathcal{G}^r)_{r=a}^{b}$ but neither of $(\mathcal{G}^r)_{r=a-1}^{b}$ nor $(\mathcal{G}^r)_{r=a}^{b+1}$.*

Finally, a graph sequence that has a unique common root is called a single-rooted sequence.

Definition 3 (Single-rooted sequence). *We call a sequence $(\mathcal{G}^r)_{r\in I}$ single-rooted, or R-single-rooted, if there*

[5]In [3,4], the term *I-vertex-stable root component* (I-VSRC, or alternatively d-VSRC) has been coined for R being a common root in $(\mathcal{G}^r)_{r\in I}$ with $I = [a, a + d - 1]$. We prefer the more general term common root of a sequence in this paper, since it aligns better with the focus of our analysis on (possibly arbitrary) sequences of communication graphs.

exists a unique root component R s.t. $\forall i,j \in I : \text{roots}(\mathcal{G}^i) = \text{roots}(\mathcal{G}^j) = \{R\}$. We call R a maximal single root of a consecutive graph sequence $(\mathcal{G}^r)_{r\in I}$ with $I = [a,b]$, iff R is a single root of $(\mathcal{G}^r)_{r=a}^{b}$ but neither of $(\mathcal{G}^r)_{r=a-1}^{b}$ nor $(\mathcal{G}^r)_{r=a}^{b+1}$.

We now introduce a notion of *causal past*, which is closely related to the classic "happens-before" relation [14], albeit presented in a way that is compatible with the process-time graphs used e.g. in [12]. Given some round b, p's *causal past* $\text{CP}_p^b(a)$ down to round a are exactly those processes the state of which at the end of round a has affected the state of p at the end of round b.

DEFINITION 4 (CAUSAL PAST). *For a given infinite sequence σ of communication graphs, we define the causal past $\text{CP}_p^b(a)$ of process p from (the end of) round b down to (the end of) round a as $\text{CP}_p^b(b) := \{p\}$ and for $a < \ell \leqslant b$, $\text{CP}_p^b(\ell-1) := \text{CP}_p^b(\ell) \cup \{q \in \Pi \mid \exists q' \in \text{CP}_p^b(\ell) : (q' \to q) \in \mathcal{G}^\ell\}$*

Note carefully an important consequence of Definition 4: By definition, $q \in \text{CP}_p^b(a)$ implies that the state of q at the end of round a is in the causal past of p by the end of round b. Since the latter is a direct result of the communication graphs up to round b, however, this implies that p must have got the information about the round a state of q already *before* it performs its round b computing step, e.g., in a round b message. Thus, p can use that information already in its round b computation.

From the monotonic growth of $\text{CP}_p^b(a)$ (recall the self-loops in every $\mathcal{G}^r$), we can deduce the following corollary:

COROLLARY 2. *$p \in \text{CP}_q^b(a)$ implies $p \in \text{CP}_q^{b'}(a)$ for all $b' \geqslant b$. Analogously, $p \in \text{CP}_q^b(a)$ implies that $p \in \text{CP}_q^b(a')$ for all $a' \leqslant a$.*

As it will turn out in the next section, the "multi-hop delay" of a message sent by some process to reach some other process(es), i.e., the speed of information propagation over multiple rounds, will be important for solving consensus. This is particularly true in the case of a single-rooted graph sequence, where the following lemma guarantees an upper bound of $n-1$ rounds:

LEMMA 1. *Let σ be a graph sequence containing a sequence $S = (\mathcal{G}^{r_1}, \dots, \mathcal{G}^{r_{n-1}})$ of $n-1$ not necessarily consecutive R-single-rooted communication graphs. Then, for all $p \in \Pi : R \subseteq \text{CP}_p^{r_{n-1}}(r_1 - 1)$.*

PROOF. Pick an arbitrary process $p \in \Pi$, $q \in R$. We show by induction that, for $\ell \in [1, n-1]$, $|\text{CP}_p^{r_{n-1}}(r_{n-\ell})| \geqslant \ell$ or $q \in \text{CP}_p^{r_{n-1}}(r_{n-\ell})$. For $\ell = 1$, this follows directly from Definition 4. For the induction step, we assume that the claim holds for $\ell \in [1, n-1)$ and show that it holds for $\ell+1$ as well. If the claim holds because $q \in \text{CP}_p^{r_{n-1}}(r_{n-\ell})$, by Corollary 2, we have $q \in \text{CP}_p^{r_{n-1}}(r_{n-\ell-1})$. Thus, assume that $q \notin \text{CP}_p^{r_{n-1}}(r_{n-\ell})$ and $|\text{CP}_p^{r_{n-1}}(r_{n-\ell})| \geqslant \ell$. If it holds that $|\text{CP}_p^{r_{n-1}}(r_{n-\ell})| > \ell$, we get $|\text{CP}_p^{r_{n-1}}(r_{n-\ell-1})| \geqslant \ell+1$ immediately, so assume that $|\text{CP}_p^{r_{n-1}}(r_{n-\ell})| = \ell$. Since $\mathcal{G}^{r_{n-\ell}}$ is R-single-rooted, there is a path from q to p in $\mathcal{G}^{r_{n-\ell}}$, according to Corollary 1. Because $q \notin \text{CP}_p^{r_{n-1}}(r_{n-\ell})$, there is some process q' on the path from q to p s.t. $q' \notin \text{CP}_p^{r_{n-1}}(r_{n-\ell})$ but $(q' \to p') \in \mathcal{G}^{r_{n-\ell}}$ for some $p' \in \text{CP}_p^{r_{n-1}}(r_{n-\ell})$. By Definition 4, $\text{CP}_p^{r_{n-1}}(r_{n-\ell-1}) \supseteq \text{CP}_p^{r_{n-1}}(r_{n-\ell}) \cup \{q'\}$. By the induction hypothesis, therefore $|\text{CP}_p^{r_{n-1}}(r_{n-\ell-1})| \geqslant \ell+1$. □

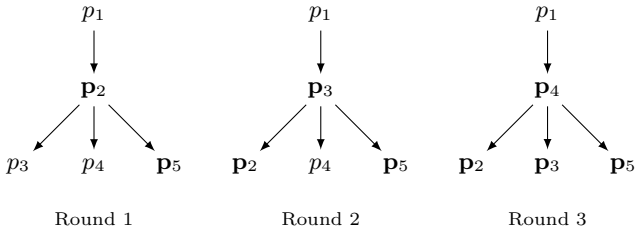

Figure 1: Example of a communication graph sequence with dynamic diameter $D = 4$, despite a small hop distance (diameter = 2) in every single graph. Bold nodes represent processes in the causal past $\text{CP}_{p_5}^3(0)$.

In order to specify message adversaries that guarantee faster information propagation than guaranteed by Lemma 1, we introduce a system parameter called *dynamic (network) diameter* $1 \leqslant D \leqslant n-1$. Intuitively, it ensures that the information from all nodes in R has reached all nodes in the network if D R-single-rooted graphs have occurred in a graph sequence.

DEFINITION 5 (DYNAMIC DIAMETER D). *A message adversary $\texttt{MA}$ guarantees a dynamic (network) diameter D, if for every graph sequence $\sigma \in \texttt{MA}$ that contains a subsequence $S = (\mathcal{G}^{r_1}, \dots, \mathcal{G}^{r_D})$ of D not necessarily consecutive R-single-rooted communication graphs, it holds that $R \subseteq \text{CP}_p^{r_D}(r_1 - 1)$ for every $p \in \Pi$.*

It was shown in [3, Theorem 3] that processes need to know some estimate of D for solving consensus: Without this knowledge, it is impossible to locally verify a necessary condition for solving consensus, namely, the ability of some process to disseminate its initial value system-wide. Note carefully, though, that knowledge of D does not permit the processes to determine n in general.

Definition 5 may lead to the conjecture that a maximum hop distance of D between $q \in R$ and $p \in \Pi$ in every $\mathcal{G}^{r_1}, \dots, \mathcal{G}^{r_D}$ guarantees a dynamic diameter of D. This is not the case, however: Consider, for example, the three-round sequence $(\mathcal{G}^r)_{r=1}^3$ of communication graphs for processes $p_1, \dots, p_5$ shown in Fig. 1. Herein, $\mathcal{G}^1$ is a directed tree of height 3, with single root node p_1 and a single node in the second level. In the following rounds, this second level node switches places with a new node $\neq p_5$ from the third level. In this scenario, $p_1 \notin \text{CP}_{p_5}^3(0)$, even though the length of the path from p_1 to any other process is $\leqslant 2$ in every $\mathcal{G}^r$.

3 A simple stabilizing message adversary

Recall that the purpose of our stabilizing message adversary is to allow an unbounded (but finite) initial period of "chaotic" behavior, where the communication graphs can be arbitrary: Unlike in [3], any $\mathcal{G}^r$ may be arbitrarily sparse and could contain several root components here. Clearly, one cannot hope to solve consensus during this initial period in general. Eventually, however, the adversary must start to generate suitably restricted communication graphs, which should allow the design of algorithms that solve consensus. Naturally, there are many conceivable restrictions and, hence, many different message adversaries that could be considered here. We will develop two instances in this paper, and also relate those to the message adversary introduced in [4].

The simple message adversary $\Diamond\texttt{STABLE}(D)$ defined in this section uses a straightforward means for closing the initial period, which is well-known from eventual-type models in distributed computing: In partially synchronous systems [7], for example, one assumes that speed and communication delay bounds hold forever from some unknown stabilization time on. Analogously, we assume that there is some unknown round r_{stab}, from which on the adversary must behave "nicely" forever. Albeit the resulting message adversary is restricted in its behavior, it provides easy comparability of the performance (in particular, of the termination times) of different consensus algorithms. Moreover, in Section 6, we will show how to generalize $\Diamond\texttt{STABLE}(D)$ to a considerably stronger message adversary $\Diamond\texttt{STABLE}'(D)$, which does not require such a restrictive "forever after" property.

In order to define what "behaving nicely" actually means in the case of $\Diamond\texttt{STABLE}(D)$, we start from a necessary condition for solving consensus in $(\mathcal{G}^r)_{r=r_{\text{stab}}}^{\infty}$: The arguably most obvious requirement here is information propagation from a non-empty set of processes to all processes in the system. According to Lemma 1, this can be guaranteed when there is a sufficiently long sub-sequence of communication graphs in $(\mathcal{G}^r)_{r=r_{\text{stab}}}^{\infty}$ with a single common root. Natural candidate choices for feasible graphs would hence be the very same single-rooted graph $\mathcal{G}$ in all rounds $r \geqslant r_{\text{stab}}$, or the assumption that all $\mathcal{G}^r$ are strongly or even completely connected (and hence also single-rooted). While simple, these choices would impose severe and unnecessary restrictions on our message adversary, however, which are avoided by the following more general definition (that includes these choices as special instances, and hence results in a stronger message adversary):

Definition 6. *We say that $(\mathcal{G}^r)_{r=1}^{\infty}$ has a (unique)* FAES*-common root R ("forever after, eventually single") starting at round $r_{stab} \geqslant 1$, iff R is (i) a maximal common root of $(\mathcal{G}^r)_{r=r_{stab}}^{\infty}$ and (ii) a maximal single root of $(\mathcal{G}^r)_{r=r_{sr}}^{\infty}$, for some round $r_{sr} \geqslant r_{stab}$.*

$\Diamond\texttt{STABILITY}$ contains those communication graph sequences $(\mathcal{G}^r)_{r=1}^{\infty}$ that have a FAES*-common root R.*

Note that the eventual single-rootedness of $(\mathcal{G}^r)_{r=r_{\text{stab}}}^{\infty}$ implied by $\Diamond\texttt{STABILITY}$ allows the respective round graphs $\mathcal{G}^r$ to be very sparse: For instance, each $\mathcal{G}^r$ of $(\mathcal{G}^r)_{r=r_{\text{stab}}}^{\infty}$ consisting of a chain with the same head but varying body would satisfy the requirement for single-rootedness.

Whereas the properties guaranteed by $\Diamond\texttt{STABILITY}$ will suffice to ensure liveness of the consensus algorithm presented in Section 5, i.e., termination, it is not sufficient for also ensuring safety, i.e., agreement. Consider for instance the top run (execution ε_1) from Fig. 2, where p is connected to q in a chain forever, which is feasible for $\Diamond\texttt{STABILITY}$. In any correct solution algorithm, the head p of this chain must eventually decide in some round τ on its initial value x_p. Now consider the execution ε_2, depicted in the bottom of Fig. 2, where p is disconnected until τ and $x_p \neq x_q$. Since ε_2 is indistinguishable for p from ε_1 until τ, process p will decide x_p at time τ. However, in ε_2, a chain forms with head $q \neq p$ forever after τ. Since q is only aware of its own input value x_q, it can never make a safe decision in this execution.

This is why $\Diamond\texttt{STABLE}(D)$ needs to combine $\Diamond\texttt{STABILITY}$ with another message adversary $\texttt{STICKY}(x)$ that enables our solution algorithm to also ensure safety. The above example illustrates the main problem that we face here: If we allow

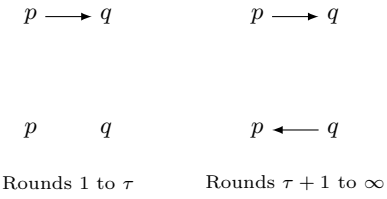

Figure 2: Two executions ε_1 (top) and ε_2 (bottom), indistinguishable for p until τ.

root components to remain common for too many consecutive rounds in the initial period (before r_{stab}), the members of such a root component (which does not need to be single) cannot distinguish this from the situation where they are belonging to the final FAES-common root (after r_{stab}). In [3], this problem was void since *all* communication graphs were assumed to be single-rooted. In the following Definition 7, we require that every root R that is common during a sequence of "significant" length $x+1$ is already the FAES-common root R. Again, in Section 6, we will present a significant relaxation of this quite restrictive (but convenient) assumption.

Definition 7. $\texttt{STICKY}(x)$ *contains those communication graph sequences $\sigma = (\mathcal{G}^r)_{r=1}^{\infty}$, where every root R that is common for $> x$ consecutive rounds in σ is the* FAES*-common root R in σ.*

We are now ready to define our simple eventually stabilizing message adversary $\Diamond\texttt{STABLE}(D)$, which is the conjunction of the adversaries from Definitions 6 and 7, augmented by the additional requirement to always guarantee a dynamic network diameter D according to Definition 5:

Definition 8. *The message adversary $\Diamond\texttt{STABLE}(D) = \texttt{STICKY}(D) + \Diamond\texttt{STABILITY}$ contains those graph sequences of $\texttt{STICKY}(D) \cap \Diamond\texttt{STABILITY}$ that guarantee a dynamic diameter of D.*

For exemplary graph sequences of $\Diamond\texttt{STABLE}(D)$ with $D = 2$, see Figs. 3 and 4. Note carefully that Definition 6 allows the coexistence of the FAES-common root R with some other root component $R' \neq R$ in communication graphs that occur before R becomes the single root (in round r_{sr}). However, according to Definition 7, R' cannot be common root for more than D consecutive rounds in this case.

In the remainder of this section, we will informally introduce the message adversary $\texttt{VSRC}(n, 4D) + \texttt{MAJINF}(k)$ introduced in [4].[6] The latter paper introduced a consensus algorithm, which gracefully degrades to k-set agreement[7] in less favorable runs. $\texttt{VSRC}(n, 4D)$ consists of all graph sequences σ, where up to n root components (the maximal possible number) are allowed in every graph $\mathcal{G}^r$. In addition, there must be a consecutive subsequence of graphs $(\mathcal{G}^r)_{r=r_{\text{stab}}}^{r_{\text{stab}}+4D-1} \subseteq \sigma$ where all root components are common[8] and ensure dynamic network diameter D. On the

[6] In [4], a network diameter H and a root diameter D are distinguished; we set $H = D$ here to ensure compatibility with our definitions.

[7] In k-set agreement, the consensus agreement condition is relaxed such that up to k different decision values are permitted.

[8] Recall that these common root components are called $4D$-vertex-stable root components ($4D$-VSRCs) in [4].

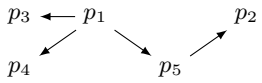

Figure 3: Execution ε of Theorem 2, $n = 5$, $D = 2$

other hand, $\texttt{MAJINF}(1)$ guarantees that the first $2D$+1-VSRC that occurs in a run dominantly influences every subsequent $2D + 1$-VRSC. This ensures that a decision value possibly generated in an earlier $2D + 1$-VRSC is duly propagated to every subsequent $2D + 1$-VSRCs. In the following Theorem 1, we show that $\texttt{VSRC}(n, 4D) + \texttt{MAJINF}(1)$ is stronger than $\Diamond\texttt{STABLE}(D)$. This implies that the consensus algorithm from [4] works also under $\Diamond\texttt{STABLE}(D)$.

THEOREM 1. *Message adversary* $\texttt{VSRC}(n, 4D)$+$\texttt{MAJINF}(1)$ *is stronger than* $\Diamond\texttt{STABLE}(D)$*, i.e.,* $\texttt{VSRC}(n, 4D)$+$\texttt{MAJINF}(1) \supseteq \Diamond\texttt{STABLE}(D)$

PROOF. Since both adversaries guarantee the dynamic diameter D, it suffices to show that $\texttt{VSRC}(n, 4D) \supseteq \Diamond\texttt{STABILITY}$ and $\texttt{MAJINF}(1) \supseteq \texttt{STICKY}(D)$ both hold.

$\texttt{VSRC}(n, 4D) \supseteq \Diamond\texttt{STABILITY}$: Take any feasible sequence σ of $\Diamond\texttt{STABILITY}$. By Definition 6, there is some round $r_{\text{sr}} \geqslant r_{\text{stab}}$ from which on $(\mathcal{G}^r)_{r=r_{\text{sr}}}^{\infty} \subseteq \sigma$ is R-single-rooted. But then also $(\mathcal{G}^r)_{r=r_{\text{sr}}}^{r_{\text{sr}}+4D-1}$ is R-single-rooted and hence $\sigma \subseteq \texttt{VSRC}(n, 4D)$.

$\texttt{MAJINF}(1) \supseteq \texttt{STICKY}(D)$: Pick an arbitrary feasible sequence σ of $\texttt{STICKY}(D)$. If there is a subsequence $(\mathcal{G}^r)_{r\in I}$ of σ with common root R consisting of $> 2D$ rounds, then it follows from Definition 7 that there cannot be a subsequence $(\mathcal{G}^r)_{r\in I'}$ of σ with common root $R' \neq R$ consisting of $> 2D$ rounds, as R and R' both would need to be the single root of $(\mathcal{G}^r)_{r=r_{\text{sr}}}^{\infty}$. Hence, σ is trivially in $\texttt{MAJINF}(1)$. □

4 Termination time lower bound

It follows immediately from Theorem 1 that the gracefully degrading consensus algorithm from [4] works also under $\Diamond\texttt{STABLE}(D)$. According to [4, Lemma 5], it terminates at the end of round $r_{\text{sr}} + 4D$, i.e., has a termination time of $4D + 1$ rounds measured from the start of the stable period (round r_{sr}).

From an applications perspective, fast termination is of course important. An interesting question is hence whether the algorithm from [4] is optimal in this respect. The following Theorem 2 provides us with a lower bound of $2D$ for the termination time under message adversary $\Diamond\texttt{STABLE}(D)$, which proves that it is not: There is a substantial gap of $2D$ rounds.

THEOREM 2. *Solving consensus is impossible under message adversary* $\Diamond\texttt{STABLE}(D)$ *in round* $r_{sr} + 2D - 1$.

PROOF. We will use a contradiction proof based on the indistinguishablility of specifically constructed admissible executions. Since the processes have no knowledge of Π and $|\Pi|$, we can w.l.o.g. assume that $n \geqslant 4$ and $D < n - 2$.

Assume that an algorithm $\mathcal{A}$ exists that solves consensus under $\Diamond\texttt{STABLE}(D)$ by the end of round $r_{\text{sr}} + 2D - 1$. Then, $\mathcal{A}$ must also solve consensus in the following execution ε: In ε, all processes in Π start with input value 0, and all graphs in $(\mathcal{G}^r)_{r=1}^{\infty}$ are the same $\mathcal{G}$. The graph $\mathcal{G}$ is single-rooted with $R = \{p_1\}$ and contains a chain $\mathcal{C} \subset \mathcal{G}$ consisting of $D + 1$ processes $C \subseteq \Pi$ that starts in $p_1 \in C$ and ends in $p_2 \in C$. All remaining processes are direct out-neighbors of p_1. Fig. 3 shows an example of the graph $\mathcal{G}$ used in ε for $n = 5$ and $D = 2$. The execution is admissible because its graph sequence is feasible for $\Diamond\texttt{STABLE}(D)$ with $r_{\text{sr}} = r_{\text{stab}} = 1$. By validity and our termination time assumption, every process must hence have decided 0 by the end of round $r_{\text{sr}} + 2D - 1$ in ε.

We will now construct an execution ε' of $\mathcal{A}$, where some process in $\Pi \setminus \{p_1, p_2\}$ eventually decides 1 albeit the state ${p_2}^{r_{\text{sr}}+2D-1}$ of process p_2 at the end of round $r_{\text{sr}} + 2D - 1$ is the same as in ε. Thus, ε and ε' are indistinguishable for process p_2 until $r_{\text{sr}} + 2D - 1$. An example of the graph sequence used in ε' for $n = 5$ and $D = 2$ is shown in Fig. 4.

In ε', let two processes $\{p_3, p_4\}$ in $\Pi \setminus C$ have initial value 1 and all remaining ones have initial value 0. The identical graph $\mathcal{G}'$ used in $(\mathcal{G}^r)_{r=1}^{D}$ consist of the very same chain $\mathcal{C}$ as in $\mathcal{G}$, and a single edge (p_3, p_4). Note that $\mathcal{G}'$ contains two root components, namely $R_1 = \{p_1\}$ and $R_2 = \{p_3\}$. The identical graph $\mathcal{G}''$ used in $(\mathcal{G}^r)_{r=D+1}^{2D}$ consist of the chain $\mathcal{C}$, an additional edge p_2 to p_1, and an edge (p_4, p_3). Again, $\mathcal{G}''$ contains two root components, $R_1 = C$ and $R_2 = \{p_4\}$. Finally, the graph $\mathcal{G}'''$ used in $(\mathcal{G}^r)_{r=2D+1}^{\infty}$ is $\mathcal{G}''$ augmented by two edges connecting p_4 to two different process in C. Note that it contains a single root $R = \{p_4\}$ and guarantees a dynamic diameter of (at most) D.

Clearly, ε' is an admissible execution for $\Diamond\texttt{STABLE}(D)$: It adheres to $\Diamond\texttt{STABILITY}$ for $r_{\text{sr}} = D + 1$, when $\{p_4\}$ becomes a forever common root that becomes single forever starting with round $2D + 1$. It is also feasible for $\texttt{STICKY}(D)$, as the only graph sequence that contains a common root for more than D rounds, namely, the final one $(\mathcal{G}^r)_{r=2D+1}^{\infty}$, is single-rooted.

For p_2, the executions ε and ε' are indistinguishable for the first $2D$ rounds, because by the end of round $2D$, p_2 cannot have learned of the existence of the edge $(p_2 \to p_1)$ that distinguishes the root components R and R_1 involving p_1 in $\mathcal{G}$ and $\mathcal{G}''$, respectively: It takes at least D rounds for any information, sent by p_1, to be forwarded along $\mathcal{C}$ to p_2, and p_1 cannot have learned about the existence of this edge before round $D + 1$. It hence follows that p_2 decides 0 in round $2D$ also in ε', as it does so in ε.

In ε', by validity and the assumed correctness of $\mathcal{A}$, however, all processes must eventually decide 1 to solve consensus: The only input value that p_4 ever gets to know throughout the entire execution is 1. The same is true in the execution ε'', which is identical to ε' except that the input value of all processes is 1. Clearly, p_4 must decide 1 in ε'' and, hence, also in ε'. This provides the required contradiction and completes our proof.

Above, we have shown the impossibility for the case where $r_{\text{sr}} = 1$ (which would already be sufficient for the claim of Theorem 2). Actually, it is not hard extend the proof for general r_{sr}, by simply prefixing ε and ε' with the following graph sequence π: In every round $\leqslant r_{\text{sr}}$ of π, the graphs alternate between $\mathcal{G}'$ and $\mathcal{G}''$, such that the graph in the last round of π is $\mathcal{G}''$. The resulting prefixed executions obviously still adhere to the message adversary $\Diamond\texttt{STABLE}(D)$ and are indistinguishable from their respective prefixed counterparts for processes p_2 and p_4. □

We will show in the next section that the lower bound

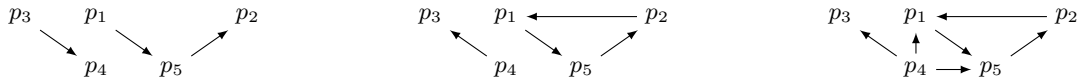

Figure 4: Execution ε' of Theorem 2, $n = 5$, $D = 2$

established in Theorem 2 is tight, by providing a matching algorithm.

5 A fast consensus algorithm

We now present our consensus algorithm for the message adversary $\Diamond\texttt{STABLE}(D)$, which also works correctly under the generalized $\Diamond\texttt{STABLE}'(D)$ that will be introduced in Section 6. The algorithm is based on the fact that, from the messages a node receives, it can reconstruct a faithful under-approximation of (the relevant part of) the communication graph of every round, albeit with delay D.

The algorithm stated in Fig. 5 works as follows: Every process p maintains an array $\widehat{\mathcal{G}}_p[r]$ that holds the *graph approximation* of $\mathcal{G}^r$, and a matrix $\texttt{lock}_p[q][r]$ that holds the history of a special value, the *lock-value*, for every known process q and every round r. $\widehat{\mathcal{G}}_p^m[r]$ and $\texttt{lock}_p^m[q][r]$ denote the content of the respective array entry at the end of round m as usual. The first entries of these arrays are initialized to the singleton-graph $\widehat{\mathcal{G}}_p^0[0] = (\{p\}, \{\})$ resp. to $\texttt{lock}_p^0[p][0] := x_p$, the input value of p, and to $\texttt{lock}_p^0[q][0] := \perp$ for every $q \neq p$. Note that $\texttt{lock}_p[p][m-1]$ can be viewed as p's *proposal value* for round m. Every process broadcasts $\widehat{\mathcal{G}}_p^{m-1}[r]$ and $\texttt{lock}_p^{m-1}[q][r]$ in round $m \geqslant 1$, and updates $\widehat{\mathcal{G}}_p^m[r]$ and $\texttt{lock}_p^m[q][r]$, by fusing the information contained in the messages received in round m in a per-round fashion (as detailed below), before executing the round m *core computation* (we will omit the attribute core in the sequel if no ambiguity arises) of the algorithm. Note that the round m core computation for $m \in \{1, \ldots, D\}$ is empty.

In the computation of some round τ, p will eventually decide on the maximum $\texttt{lock}_p[q][a]$ value for all $q \in R$, where R is a common root of some sequence $(\mathcal{G}^r)_{r=a}^{a+D}$ but not of $(\mathcal{G}^r)_{r=a-1}^{a+D-1}$, as detected locally in $\widehat{\mathcal{G}}_p^\tau[*]$. Note carefully that τ may be different for processes other than p.

Two mechanisms are central to the algorithm for accomplishing this: First, any process p that, in its round m computation, locally detects a single root component R in $\widehat{\mathcal{G}}_p^m[m-D]$ will "lock" it, i.e., assign the maximum value of $\texttt{lock}_p^m[q][m-D]$ for any $q \in R$ to $\texttt{lock}_p^m[p][m]$. Second, if process p detects in round τ that a graph sequence had a common root R' for at least $D+1$ rounds in its graph approximation, starting in round a, p will decide, i.e., set y_p to the maximum of $\texttt{lock}_p^\tau[q][a]$ among all $q \in R'$.

Informally, the reason why this algorithm works is the following: From detecting an R-single-rooted sequence of length $\geqslant D+1$, p can infer, by the $\texttt{STICKY}(D)$ property of our message adversary, that the entire system is about to lock p's decision value. Moreover, by exploiting the information propagation guarantee given by Lemma 1, we can be sure that, after p's decision in round τ, every other process q decides (in some round $\tau' \geqslant \tau$) on the very same value: Under $\Diamond\texttt{STABLE}(D)$, it decides because the root that triggered the decision of p is the FAES-common root; under $\Diamond\texttt{STABLE}'(D)$, q decides on the same value because it will never assign a value different from $\texttt{lock}_p[p][\tau]$ to $\texttt{lock}_q[x][\tau'']$ for any $\tau'' \geqslant \tau'$ and any known process x. Finally, termination is guaranteed since every p will eventually find an R-single-rooted sequence of duration at least $D+1$ because of $\Diamond\texttt{STABILITY}$.

Graph approximation and lock maintenance

Our algorithm relies on a simple mechanism for maintaining the graph approximation $\widehat{\mathcal{G}}_p[r]$ and the array of lock values $\texttt{lock}_p[q][r]$ at every process p: In every round, each process p broadcasts its current $\widehat{\mathcal{G}}_p[*]$ and $\texttt{lock}_p[*][*]$ and updates all entries with new information possibly obtained in the received approximations from other processes. In more detail, an edge $(q \to q')$ will be present in $\widehat{\mathcal{G}}_p^m[r]$ at the end of round $m \geqslant r$ if either $p = q'$ and p received a message from q in round r, or if p received $\widehat{\mathcal{G}}_{q''}^{r''}[r]$ for $m \geqslant r'' \geqslant r$ from some process q'' and $(q \to q') \in \widehat{\mathcal{G}}_{q''}^{r''}[r]$. Similarly, $\texttt{lock}_p^m[q][r]$ for $r < m$ is updated to $\texttt{lock}_{q'}[q][r] \neq \perp$ whenever such an entry is received from any process q'; the entry $\texttt{lock}_p^m[q][m]$ for the current round m is initialized to $\texttt{lock}_p[p][m] := \texttt{lock}_p[p][m-1]$ for $q = p$ and to $\texttt{lock}_p[q][m] := \perp$ for every $q \neq p$.

Note carefully that we assume that the round m computation of the approximation algorithm is executed *before* the round m core computing step at every process. Therefore, the round m approximation $\widehat{\mathcal{G}}_p^m[*]$ is already available *before* the core computing step of round m at process p is executed.

We do not provide further details of the implementation of this graph approximation here; a fitting algorithm, along with its correctness proof, can be found in [3,4]. We remark, though, that the full-information approach of the above implementation incurs sending and storing a large amount of redundant information. Comments related to a more efficient implementation are provided in Section 6.

The crucial property guaranteed by our graph approximation is that processes under-approximate the actual communication graph, i.e., that they do not fabricate edges in their approximation. Using our notion of causal past, it is not difficult to prove the following assertion about edges that are guaranteed to exist in the graph approximation:

LEMMA 2. *In a full-information graph approximation protocol, $q \in \mathrm{CP}_p^{r'}(r)$ holds for $r' > r \Leftrightarrow$ there exists a process q' s.t. $(q \to q') \in \widehat{\mathcal{G}}_p^{r'}[r'']$ for some $r'' \in (r, r']$.*

PROOF. "⇒"-direction: If $p = q$, the claim trivially holds because every communication graph contains the self-loop $(p \to p)$. For $p \neq q$, since we assume $q \in \mathrm{CP}_p^{r'}(r)$, by Definition 4, there exists a round $r'' > r$, such that $\exists q' \in \mathrm{CP}_p^{r'}(r'')$ with $(q \to q') \in \mathcal{G}^{r''}$. Therefore, p must have received the round r'' state of q' and hence learned about the edge $(q \to q')$, by round r'. In other words, $(q \to q') \in \widehat{\mathcal{G}}_p^{r'}[r'']$, as claimed.

"⇐"-direction: Since we assume a full-information protocol, p knowing part of the state of another process q' im-

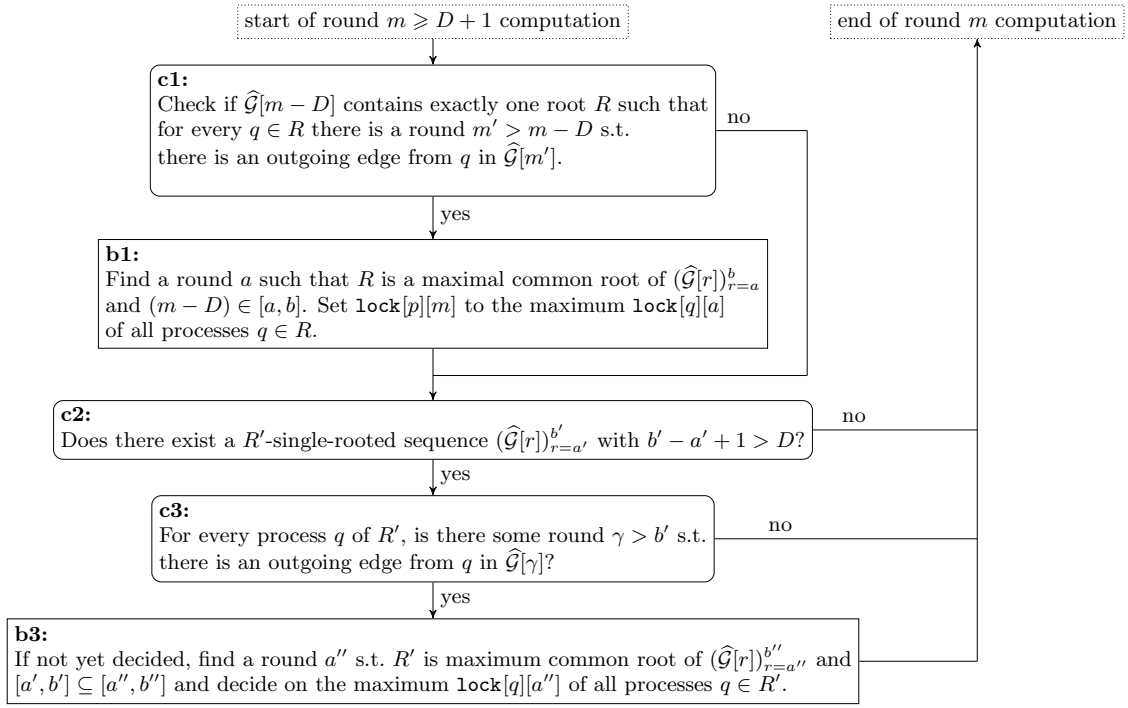

Figure 5: Round $m \geqslant D+1$ core computation step of our consensus algorithm for process p. $\widehat{\mathcal{G}}[r] = \widehat{\mathcal{G}}_p^m[r]$ denotes p's round m view of $\mathcal{G}^r$ provided by the network approximation algorithm. lock$[q][r]$ denotes lock$_p^m[q][r]$, where lock$[p][m]$ represents p's proposal value for the next round $m+1$.

plies that p knows the entire state of q'. Hence, if $(q \to q') \in \widehat{\mathcal{G}}_p^{r'}[r'']$, p knows the state of q' of round r''. Thus $q' \in \mathrm{CP}_p^{r'}(r'')$ with $r'' \leqslant r'$. From Corollary 2, it follows that $q \in \mathrm{CP}_p^{r'}(r''-1)$, which implies $q \in \mathrm{CP}_p^{r'}(r)$ because $r \leqslant r''$. □

We now present a more abstract view on this mechanism of approximating the communication graph. First, we answer which state information a process needs in order to reliably detect which roots are present in the actual communication graph.

Lemma 3. *Let $R \in \mathrm{roots}(\mathcal{G}^r)$ and let there be some process p and round r' such that $R \subseteq \mathrm{CP}_p^{r'}(r)$. In a full-information graph approximation protocol, $R \in \mathrm{roots}(\widehat{\mathcal{G}}_p^{r'}[r])$. Furthermore, there exists a process q' s.t. $(q \to q') \in \widehat{\mathcal{G}}_p^{r'}[r'']$ for some $r < r'' \leqslant r'$.*

Proof. Since $R \subseteq \mathrm{CP}_p^{r'}(r)$, according to Corollary 2, by the end of round r', p has received the round r state q^r of all processes $q \in R$. In particular, p has received all round r in-edges of every process q. Hence, R is a strongly connected component of $\widehat{\mathcal{G}}_p^{r'}[r]$ and there are no processes $q' \in \Pi \setminus R$ s.t. $(q' \to q) \in \widehat{\mathcal{G}}_p^{r'}[r]$. But then, $R \in \mathrm{roots}(\widehat{\mathcal{G}}_p^{r'}[r])$, as asserted. The presence of $(q \to q')$ in $\widehat{\mathcal{G}}_p^{r'}[r'']$ follows directly from Lemma 2. □

We conclude our considerations regarding the graph approximation by looking at what is sufficient from an algorithmic point of view for a process p to faithfully determine the root components in some communication graph. In the case where a root component $R \in \mathrm{roots}(\mathcal{G}^r)$ has size $|R| > 1$, we note that as soon as a process p knows, in some round r', at least one in-edge $(q' \to q) \in \widehat{\mathcal{G}}_p^{r'}[r]$ for each $q \in R$, then p knows q^r and hence all in-edges of q. Consequently, it can reliably deduce that indeed $R \in \mathrm{roots}(\mathcal{G}^r)$.

In the case where $|R| = |\{q\}| = 1$, if p has no edge $(q' \to q) \in \widehat{\mathcal{G}}_p^{r'}[r]$, this is *not* sufficient for concluding that $\{q\} \in \mathrm{roots}(\mathcal{G}^r)$: Process p seeing no in-edge to a process q in the local graph approximation $\widehat{\mathcal{G}}_p^{r'}[r]$ happens naturally if $q \in \mathrm{CP}_p^{r'}(r-1)$ and $q \notin \mathrm{CP}_p^{r'}(r)$, i.e., when the last message p received from q was sent at the beginning of round r. In order to overcome this issue, process p must somehow ascertain that it already received the state q^r of process q in round r. In particular, process p can deduce this directly from its graph approximation as soon as it observed some outgoing edge from q in a round strictly after r.

Let us state this more formally in the following lemma.

Lemma 4. *Consider a full-information graph approximation protocol. Let $R \in \mathrm{roots}(\widehat{\mathcal{G}}_p^{r'}[r])$ for $r' > r$, and let, for all processes $q \in R$, there be a process q' and a round $r'' \in (r, r']$, such that $(q \to q') \in \widehat{\mathcal{G}}_p^{r'}[r'']$. Then, $R \in \mathrm{roots}(\mathcal{G}^r)$, and $R \subseteq \mathrm{CP}_p^{r'}(r)$.*

Proof. By contradiction. Assume that $R \in \mathrm{roots}(\widehat{\mathcal{G}}_p^{r'}[r])$, $\forall q \in R \exists q' \in \Pi, r'' \in (r, r']\colon (q \to q') \in \widehat{\mathcal{G}}_p^{r'}[r'']$ and $R \notin \mathrm{roots}(\mathcal{G}^r)$. Because of the latter, there exist some processes $q \in R$ and $q'' \notin R$ with $(q'' \to q) \in \mathcal{G}^r$. By the presence of the edge $(q \to q')$ in $\widehat{\mathcal{G}}_p^{r'}[r'']$ and Lemma 2, we have $R \subseteq \mathrm{CP}_p^{r'}(r)$. But then, by the assumption that $(q'' \to q) \in \mathcal{G}^r$, it must also hold that $(q'' \to q) \in \widehat{\mathcal{G}}_p^{r'}[r]$. This, however, contradicts that $R \in \mathrm{roots}(\widehat{\mathcal{G}}_p^{r'}[r])$. □

Finally, the way how the lock arrays are maintained by our algorithm implies the following simple results:

COROLLARY 3. *If $r' > r$, then $q \in \mathrm{CP}_p^{r'}(r)$ implies that also $\mathtt{lock}_p^{r'}[q][r''] = \mathtt{lock}_q^{r''}[q][r'']$ for all rounds $r'' \leqslant r$.*

LEMMA 5. *Let m be a round reached by process p in the execution. Then, $\mathtt{lock}_p^m[p][r] \neq \bot$ for all $0 \leqslant r \leqslant m$.*

PROOF. Since $\mathtt{lock}_p^0[p][0] = x_p$, it follows from the update rule $\mathtt{lock}_p[p][m] := \mathtt{lock}_p[p][m-1]$ that $\mathtt{lock}_p[p][m] \neq \bot$ for all reached rounds m, provided that the core algorithm never assigns $\bot$ in b1. Since the latter can only assign the maximum of $\mathtt{lock}_p[q][a]$ for all $q \in R$ from some earlier round $a \leqslant m - D < m$, the statement of our lemma follows from a trivial induction based on Corollary 3, provided we can guarantee $q \in \mathrm{CP}_p^m(a)$. The latter follows immediately from c1 in conjunction with Lemma 4, however. □

Correctness proof

Before proving the correctness of the algorithm given in Fig. 5 (Theorem 3 below), we first establish two technical lemmas: Lemma 6 reveals that our algorithm terminates for every message adversary MAT that guarantees certain properties (without guaranteeing agreement, though). The complementary Lemma 7 shows that our algorithm ensures agreement (without guaranteeing termination, though) for every message adversary MAA that guarantees certain other properties. Theorem 3 will then follow from the fact that $\Diamond\mathtt{STABLE}(D) \subseteq \mathtt{MAT} \cap \mathtt{MAA}$.

LEMMA 6. *The algorithm terminates by the end of round τ under any message adversary MAT that guarantees dynamic diameter D in conjunction with the following properties: For every $\sigma \in \mathtt{MAT}$,*

- *there is an R-single-rooted sequence $(\mathcal{G}^r)_{r=\alpha}^{\beta} \in \sigma$ with $\beta - \alpha + 1 > D$.*
- *there is a round τ such that $R \subseteq \mathrm{CP}_p^\tau(\beta)$, for all $p \in \Pi$.*

PROOF. We show that if process p has not decided before round τ, it will do so in round τ. By round τ, every process $p \in \Pi$ received q^β for all $q \in R$ by the assumption that $R \subseteq \mathrm{CP}_p^\tau(\beta)$. Hence, by Lemma 3 and Lemma 4, for every $p \in \Pi$, it holds that R is the single root of $\mathrm{roots}(\widehat{\mathcal{G}}_p^\tau[\beta])$. Furthermore, by Corollary 2, R is in fact the single root of $\mathrm{roots}(\widehat{\mathcal{G}}_p^\tau[r])$ for any $r \in [\alpha, \beta]$. Therefore, process p will pass the check c2 in round τ.

In addition, by the assumption that $R \subseteq \mathrm{CP}_p^\tau(\beta)$ and Lemma 3, for every $q \in R$, there exists a round $\beta' \in (\beta, \tau]$, s.t. $(q \to q') \in \widehat{\mathcal{G}}_p^\tau[\beta']$ for some process q'. Therefore, process p will pass the check c3 in round τ and decide. □

Lemma 7 below shows that, under message adversaries that guarantee a $\mathtt{ECS}(D+1)$-common root according to Definition 9, the algorithm from Fig. 5 satisfies agreement.

DEFINITION 9. *We say that a graph sequence $(\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$ has a $\mathtt{ECS}(x+1)$-common root ("embedded $x+1$-consecutive single common root") R, if (i) $(\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$ has a common root R and (ii) $(\mathcal{G}^r)_{r=\alpha'}^{\alpha'+x} \subseteq (\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$ has a single root R.*

LEMMA 7. *Let MAA be a message adversary that guarantees, for every $\sigma \in \mathtt{MAA}$, a dynamic diameter D in conjunction with the property that the first subsequence $(\mathcal{G}^r)_{r=\alpha}^{\beta} \subseteq \sigma$ with a maximum common root R and $\beta - \alpha + 1 > D$ has a $\mathtt{ECS}(D+1)$-common root. Under MAA, if two or more processes decide in our algorithm, then they decide on the same value $\neq \bot$.*

PROOF. Let α' and β', with $\beta' - \alpha' + 1 > D$, delimit the maximal period where R is single-rooted, as predicted by Definition 9.

Setting $\lambda = \max_{q \in R} \mathtt{lock}_q^\alpha[q][\alpha]$, we show that if an arbitrary process p decides in round τ, it decides on λ and $\lambda \neq \bot$. Assume that p decides in some round τ. It follows from c2 and c3 that p detects in round τ that R' is the single root of $(\widehat{\mathcal{G}}_p^\tau[r])_{r=a'}^{b'}$ with $b' - a' + 1 > D$, and that, for every $q \in R'$, there is a round $\gamma > b'$ where there is an edge (q, q') in $\widehat{\mathcal{G}}_p^\tau[\gamma]$ for some process $q' \in \Pi$. By Lemma 4, we have that $R' \in \mathrm{roots}(\mathcal{G}^r)$ for all $r \in [a', b']$, and $R' \subseteq \mathrm{CP}_p^\tau(b')$. Thus, Corollary 3 in conjunction with Lemma 5 confirm that indeed $\lambda \neq \bot$. We distinguish two cases:

Case 1. $[a', b'] \subseteq [\alpha, \beta]$: From the definition of MAA, in combination with the fact that $b' - a' + 1 > D$, it follows that $R' = R$: if this was not the case, then either $(\mathcal{G}^r)_{r=\alpha}^{\beta}$ would not be the first sequence of its kind or $(\mathcal{G}^r)_{r=\alpha'}^{\beta'}$ would not be R-single-rooted.

By b3, p will decide on the maximum of $\mathtt{lock}_p[q][a'']$, where a'' is a round such that $(\widehat{\mathcal{G}}_p^\tau[r])_{r=a''}^{b''}$ has a maximum common root R, $[a'', b''] \supseteq [a', b']$, and $q \in R$. Hence, since $R \subseteq \mathrm{CP}_p^\tau(b')$ and $\alpha < b'$, it follows from Corollary 2 that $R \subseteq \mathrm{CP}_p^\tau(\alpha)$. Thus, by Lemma 3, we have $a'' = \alpha$. According to Corollary 2 in conjunction with Corollary 3, it follows that p indeed decides on λ.

Case 2. $[a', b'] \not\subseteq [\alpha, \beta]$: First, observe that $a' > \beta'$: If $a' \leqslant \beta'$ then, because $(\mathcal{G}^r)_{r=\alpha}^{\beta}$ is the first sequence of its kind, we have that $a' \geqslant \alpha$. Thus, since $\mathcal{G}^{\beta'}$ is R-single-rooted, $R' = R$, and hence $[a', b'] \not\subseteq [\alpha, \beta]$ is a contradiction to the assumption that R is maximal common in $(\mathcal{G}^r)_{r=\alpha}^{\beta}$.

It follows from this observation and b3 that p decides on the maximum value of $\mathtt{lock}_p[q][a'']$ for $q \in R'$, where $a'' > \beta'$. Thus, to conclude our proof, it suffices to show that $\mathtt{lock}_p^r[p][r] = \lambda$ for all rounds $r > \beta'$ and all processes $p \in \Pi$.

Since $(\mathcal{G}^r)_{r=\beta'-D}^{\beta'}$ is R-single-rooted, it follows from Definition 5 and Lemma 3 that in round β' every process p sets $\mathtt{lock}_p^{\beta'}[p][\beta']$ to λ via b1. Moreover, if a process assigns a value to $\mathtt{lock}_p[p][m]$ during some round $m \in (\beta', \beta' + D]$ via b1 later on, it follows from the single-rootedness of $(\mathcal{G}^r)_{r=\beta'-D}^{\beta'}$ and Lemma 4 that the assigned value is also λ.

For $\ell \geqslant \beta' + D$, we show by induction on ℓ that λ is assigned to $\mathtt{lock}_p[p][m]$ (if there is any assignment at all), in round m, for all $m \in [\beta', \ell]$ and all processes p. The induction basis is $\ell = \beta' + D$, for which the claim has been established already. For the induction step, assume that the claim holds for the interval $[\beta', \ell]$ and all p. If no process p changes its lock value in b1 during the core round $\ell + 1$ computation, i.e., $\mathtt{lock}_p^\ell[p][\ell] = \mathtt{lock}_p^{\ell+1}[p][\ell+1]$, then the claim follows immediately from the induction hypothesis. Thus, assume that $\lambda = \mathtt{lock}_p^\ell[p][\ell] \neq \mathtt{lock}_p^{\ell+1}[p][\ell+1]$. This means that p has successfully passed c1 and hence, by Lemma 4, that there is a root $R'' \in \mathrm{roots}(\mathcal{G}^{\ell+1-D})$ with $R'' \subseteq \mathrm{CP}_p^{\ell+1}(\ell + 1 - D)$. If $R'' = R$ is a maximal common root of $(\mathcal{G}^r)_{r=\alpha}^{\beta}$, by Corollary 2, it follows from the definition of λ and Corollary 3 that p assigns $\mathtt{lock}_p^{\ell+1}[p][\ell+1] := \lambda$. Therefore, assume that this is not the case, i.e., $R'' \neq R$.

Still, R'' must be a maximal common root in $(\mathcal{G}^r)_{r=\alpha''}^{\beta''}$ for some $\alpha'' > \beta'$ with $\alpha'' \leqslant \ell + 1 - D$. By the induction hypothesis, $\texttt{lock}_q^{\ell+1-D}[q][r] = \lambda$ for every process q of R'' and round $r \in [\beta', \ell]$ and so, in particular, $\texttt{lock}_q^{\ell+1-D}[q][\alpha''] = \lambda$. It follows from Corollary 3 and $R'' \subseteq \mathrm{CP}_p^{\ell+1}(\ell + 1 - D)$ that for all processes $q \in R''$, we have $\texttt{lock}_p^{\ell+1}[q][\alpha''] = \lambda$. Therefore, since, by b1, p chooses its new value for $\texttt{lock}_p^{\ell+1}[p][\ell+1]$ as the maximum of the entries $\texttt{lock}_p^{\ell+1}[q][\alpha'']$, it assigns $\texttt{lock}_p^{\ell+1}[p][\ell + 1] := \lambda$. □

THEOREM 3. *The algorithm from Fig. 5 solves consensus by round $r_{sr} + 2D$ under message adversary $\Diamond\texttt{STABLE}(D)$.*

PROOF. According to b3, a process p can decide only on a value in $\texttt{lock}_p^m[p][*]$ in some round m. By Lemma 5, this value must be $\neq \perp$. Since $\texttt{lock}_q[q][0]$ is initialized to x_q for any process q, and the only assignments $\neq \perp$ to any $\texttt{lock}_q$ entry are $\texttt{lock}_{q'}$ entries of other processes, validity follows.

For agreement, recall that $\texttt{STICKY}(D)$ guarantees that the first sequence $(\mathcal{G}^r)_{r\in I}$ with a common root R and $|I| > D$ must be the $\texttt{FAES}$-common root. Hence, agreement follows from Lemma 7.

For termination, recall that $\Diamond\texttt{STABILITY}$ guarantees the existence of some round $r_{\mathrm{sr}} \geqslant r_{\mathrm{stab}}$ such that $(\mathcal{G}^r)_{r=r_{\mathrm{sr}}}^{\infty}$ is R-single-rooted. This implies that the sequence $(\mathcal{G}^r)_{r=r_{\mathrm{sr}}}^{r_{\mathrm{sr}}+D}$ is R-single-rooted and, by Definition 5, $R \subseteq \mathrm{CP}_p^{r_{\mathrm{sr}}+2D}(r_{\mathrm{sr}} + D)$. Lemma 6 thus implies termination by round $r_{\mathrm{sr}} + 2D$. □

6 Generalized stabilizing message adversary

The simple message adversary introduced in Section 3 may be criticized due to the fact that the first root component R that is common in at least $D + 1$ consecutive rounds must already be the $\texttt{FAES}$-common root that persists forever after. In this section, we will considerably relax this assumption, which is convenient for analysis and comparison purposes but maybe unrealistic in practice.

In the following Definition 10, we start with a significantly relaxed variant $\Diamond\texttt{STABILITY}'(x)$ of $\Diamond\texttt{STABILITY}$ from Definition 6: Instead of requesting an infinitely stable $\texttt{FAES}$-common root R, we only require R to be (i) a $\texttt{ECS}(x + 1)$-common root that starts at r_{stab} and becomes single at $r_{\mathrm{sr}} \geqslant r_{\mathrm{stab}}$, and (ii) to re-appear as a single root in at least D not necessarily consecutive later round graphs $\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_D}$. Note that, according to Definition 5, the latter condition ensures $R \subseteq \mathrm{CP}_p^{r_D}(r_{\mathrm{sr}} + x)$ for all $p \in \Pi$ if $\Diamond\texttt{STABILITY}'(x)$ adheres to the dynamic diameter D.

DEFINITION 10. *Every communication graph sequence $\sigma \in \Diamond\texttt{STABILITY}'(x)$ contains a subsequence $(\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$, which has a $\texttt{ECS}(x + 1)$-common root R; let $r_{stab} = \alpha$ be its starting round and $r_{sr} = \alpha'$ be the time when it becomes single. Furthermore, there are at least D, not necessarily consecutive, R-single rooted round graphs $\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_D}$ with $r_{sr} + x < r_1 < \cdots < r_D$ in σ.*

Moreover, we also relax the $\texttt{STICKY}(x)$ condition in Definition 7 accordingly: We only require that the first root component R that is common for at least $x + 1$ consecutive rounds in a graph sequence $\sigma = (\mathcal{G}^r)_{r=1}^{\infty}$ is a $\texttt{ECS}(x + 1)$-common root:

DEFINITION 11. *For every $\sigma \in \texttt{STICKY}'(x)$, it holds that the earliest subsequence in σ with a maximal common root R in at least $x+1$ consecutive rounds actually has a $\texttt{ECS}(x+1)$-common root.*

Combining these two definitions results in the following strong version of our stabilizing message adversary.

DEFINITION 12. *The strong stabilizing message adversary $\Diamond\texttt{STABLE}'(D) = \texttt{STICKY}'(D) + \Diamond\texttt{STABILITY}'(D)$ contains all graph sequences in $\texttt{STICKY}'(D) \cap \Diamond\texttt{STABILITY}'(D)$ that guarantee a dynamic diameter of D.*

Note carefully that the very first $\texttt{ECS}(D+1)$-common root R' occurring in $\sigma \in \Diamond\texttt{STABLE}'(D)$ need *not* be the $\texttt{ECS}(D+1)$-common root R guaranteed by Definition 10.

The following Lemma 8 shows that the message adversary $\Diamond\texttt{STABLE}'(D)$ is indeed weaker than $\Diamond\texttt{STABLE}(D)$. This is not only favorable in terms of model coverage, but also ensures that an algorithm designed for $\Diamond\texttt{STABLE}'(D)$ works under $\Diamond\texttt{STABLE}(D)$ as well.

LEMMA 8. $\Diamond\texttt{STABLE}(D) \subseteq \Diamond\texttt{STABLE}'(D)$

PROOF. Pick any graph sequence $\sigma \in \Diamond\texttt{STABLE}(D)$. Since $\sigma \in \Diamond\texttt{STABILITY}$, there exists a round $r_{\mathrm{sr}} \geqslant r_{\mathrm{stab}}$ such that $(\mathcal{G}^r)_{r=r_{\mathrm{sr}}}^{\infty}$ is R-single-rooted. But then $(\mathcal{G}^r)_{r=r_{\mathrm{sr}}}^{r_{\mathrm{sr}}+D}$ is also R-single-rooted and there is a set of D additional communication graphs $S = \left\{\mathcal{G}^{r_{\mathrm{sr}}+D+1}, \ldots, \mathcal{G}^{r_{\mathrm{sr}}+2D}\right\}$ such that every $\mathcal{G}^r \in S$ is also R-single-rooted. Hence, σ satisfies $\Diamond\texttt{STABILITY}'(D)$.

Furthermore, σ satisfies $\texttt{STICKY}(D)$. Thus, for the first sequence $(\mathcal{G}^r)_{r=a}^{a+D}$ with common root R, R must already be the $\texttt{FAES}$-common root and hence $(\mathcal{G}^r)_{r=r_{\mathrm{sr}}}^{\infty}$ is R-single rooted for some $r_{\mathrm{sr}} \geqslant a$. Consequently, R is a $\texttt{ECS}(x + 1)$-common root starting at a. Hence, σ satisfies $\texttt{STICKY}'(D)$. □

The following Theorem 4 shows that the algorithm from Fig. 5 also solves consensus under the stronger message adversary $\Diamond\texttt{STABLE}'(D)$:

THEOREM 4. *For a graph sequence $\sigma \in \Diamond\texttt{STABLE}'(D)$, let $\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_D}$ with $r_1 > r_{sr} + D$ denote the D re-appearances of the $\texttt{ECS}(D+1)$-common root R guaranteed by $\Diamond\texttt{STABILITY}'$ according to Definition 10. Then, the algorithm from Fig. 5 correctly terminates by the end of round $\tau = r_D$.*

PROOF. The proof of validity in Theorem 3 is not affected by changing the message adversary.

For the agreement condition, recall that $\texttt{STICKY}'(D)$ guarantees that the first sequence $(\mathcal{G}^r)_{r\in I}$ with common root R in $D+1$ consecutive rounds has a $\texttt{ECS}(D+1)$-common root. Hence, we can again apply Lemma 7 to prove that the algorithm satisfies agreement.

For the termination condition, recall that for any sequence $\sigma \in \Diamond\texttt{STABILITY}'(D)$ it is guaranteed that there exists some round r_{sr} s.t. $(\mathcal{G}^r)_{r=r_{\mathrm{sr}}}^{r_{\mathrm{sr}}+D}$ is R-single-rooted. Furthermore, σ contains at least D not necessarily subsequent R-single rooted communication graphs after $r_{\mathrm{sr}} + D$. The latter implies, by Definition 5, that $R \subseteq \mathrm{CP}_p^{\tau}(r_{\mathrm{sr}} + D)$ for every process $p \in \Pi$. Hence, we can again apply Lemma 6, which shows that the algorithm indeed terminates by round τ. □

By contrast, the algorithm from [4] does not work under $\Diamond\texttt{STABLE}'(D)$. Under an appropriate adversary, this algorithm ensures graceful degradation from consensus to general k-set agreement. This does not allow the algorithm to

adapt to the comparably shorter and weaker stability periods of $\Diamond\texttt{STABLE}'(D)$, however. In more detail, $\texttt{VSRC}(n, 4D)$ requires a four times longer period of consecutive stability than $\Diamond\texttt{STABILITY}'(D)$. The adversarial restriction $\texttt{MAJINF}(k)$ that enables k-agreement under partitions in [4] for $k > 1$, on the other hand, is very weak and thus requires quite involved algorithmic solutions. Nevertheless, despite its weakness, it is not comparable to $\texttt{STICKY}'(D)$.

Impossibility results and lower bounds

The proof of Theorem 4 indicates that two things are needed in order to solve consensus under a message adversary like $\Diamond\texttt{STABLE}'(D)$: There must be some subsequence with a single root component R in at least $x+1$ rounds, and, for every process in the system, there must be some round r such that R appears in the causal past $\mathrm{CP}_p^{r_{\mathrm{stab}}+x}(r)$. Looking more closely at the message adversary $\Diamond\texttt{STABLE}'(D)$, it is hence tempting to further weaken it by instantiating $\texttt{STICKY}'(x)$ with some $x > D$ and/or $\Diamond\texttt{STABILITY}'(x)$ with some $x < D$. There is, however, a fundamental relation between the $\texttt{STICKY}'(x)$ and $\Diamond\texttt{STABILITY}'(x)$ conditions: Weakening one condition requires strengthening the other, and vice-versa.

To further explore this issue, we introduce the message adversary $\texttt{MA}(x, y)$, which consists of the graph sequences in $\texttt{STICKY}'(x) \cap \Diamond\texttt{STABILITY}'(y)$ that guarantee a dynamic diameter D. The following Theorem 5 reveals that solving consensus requires $y \geqslant x$.

THEOREM 5. *Solving consensus is impossible under message adversary* $\texttt{MA}(x, y)$ *for* $x > y$.

PROOF. Since the processes have no knowledge of Π and $|\Pi|$, we can again w.l.o.g. assume that $n \geqslant 4$ and $D < n-2$.

Assume for a contradiction that some algorithm $\mathcal{A}$ exists that solves consensus under $\texttt{MA}(x, y)$ for $x > y$, and hence also in the following execution ε with graph sequence σ: Every process starts with input value 0 and, for the first $x \geqslant y+1$ rounds, $(\mathcal{G}^r)_{r=1}^{x}$ is R-single rooted. Then, the communication graphs alternate between being R'-single-rooted and R-single-rooted for some root $R' \neq R$. Additionally, there are two distinct processes p and q that have only incoming edges throughout the entire execution ε. The actual communication graphs outside R are such that σ has a dynamic diameter D.

Since every $\mathcal{G}^r$ in σ is single-rooted, the latter is feasible for $\Diamond\texttt{STABILITY}'(y)$, with $r_{\mathrm{stab}} = 1$ and the communication graphs $\mathcal{G}^{x+2}, \mathcal{G}^{x+4}, \ldots, \mathcal{G}^{x+2D}$ where R re-appears D times. In addition, as σ does not contain any root component that is common in more than x rounds, it trivially satisfies $\texttt{STICKY}'(x)$ as well. By the assumed correctness of $\mathcal{A}$ under $\texttt{MA}(x, y)$, there is hence some round τ by which every process must have terminated correctly.

Now consider the following execution ε', with graph sequence σ': Each process of $\Pi \setminus \{p, q\}$ starts with input value 0, while p and q start with 1. For every $\mathcal{G}^r$ of $(\mathcal{G}^r)_{r=1}^{\tau}$ in σ', the induced subgraph of $\Pi \setminus \{p, q\}$ is the same as in σ. By contrast, the processes p and q are now connected only with each other: There is an edge (q, p) in every $\mathcal{G}^r$ and an edge (p, q) in every $\mathcal{G}^r$ where r is even. Finally, the graph sequence $(\mathcal{G}^r)_{r=\tau+1}^{\infty}$ forever repeats the star-graph $\mathcal{G}$, where the center p has no in-edges and an out-edge to every other process.

Clearly, σ' is feasible for $\Diamond\texttt{STABILITY}'(y)$, with $r_{\mathrm{stab}} = \tau + 1$ due to the star-graph sequence $(\mathcal{G}^r)_{r=\tau+1}^{\infty}$. Moreover, $(\mathcal{G}^r)_{r=\tau+1}^{\infty}$ is the only subsequence of σ' with a common root R and a longer consecutive duration than x. Since R is a $\texttt{ECS}(x+1)$-common root of $(\mathcal{G}^r)_{r=\tau+1}^{\infty}$, σ' is feasible for $\texttt{STICKY}'(x)$. Since also the dynamic diameter D is adhered to in σ', we have thus that σ' is feasible for $\texttt{MA}(x, y)$.

Observe that all processes of $\Pi \setminus \{p, q\}$ have the same state in both ε and ε' at the end of round τ. Hence, all decide 0 in ε' as they do in ε. For p and q, ε' is indistinguishable from the execution ε'', which applies σ' to the initial configuration where every process started with input value 1. Consequently, p cannot make a safe decision in ε': If it decides 1, it violates agreement w.r.t. ε, if it decides 0, it violates validity w.r.t. ε''. This contradicts the assumption that $\mathcal{A}$ is a correct consensus algorithm. □

Essentially, the proof of Theorem 5 exploited the observation that the members of a root component R cannot distinguish whether they belong to the single root component guaranteed by $\Diamond\texttt{STABILITY}'(y)$ after r_{stab}, or to a (possibly non-single) "spurious" common root in $y+1$ consecutive rounds generated my $\texttt{MA}(x, y)$ before r_{stab}. Note that this is closely related to the argument used for defending the need to introduce $\texttt{STICKY}(x)$ in Definition 7 (recall the graphs depicted in Fig. 2).

In the light of Theorem 5, $\Diamond\texttt{STABLE}'(D)$ is hence the strongest eventually stabilizing variant of $\texttt{MA}(x, y)$ for $x \geqslant D$ we can hope to find an algorithm for. Note that it would not be difficult to adopt the algorithm introduced in Fig. 5 to work under $\texttt{MA}(x, y)$ for general $y \geqslant x \geqslant D$, though. Answering the question of whether it is possible to solve consensus for $x < D$ is a topic of future research.

Finally, Theorem 6 provides a termination time lower bound for consensus under $\Diamond\texttt{STABLE}'(D)$. The result itself is actually a direct consequence of the fact that $\Diamond\texttt{STABLE}(D) \subseteq \Diamond\texttt{STABLE}'(D)$ (Lemma 8) and Theorem 2. We now provide a more involved argument showing that the result holds even for arbitrary choices of r_{sr} and $\{r_1, \ldots, r_D\}$.

THEOREM 6. *For a graph sequence* $\sigma \in \Diamond\texttt{STABLE}'(D)$*, let* $\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_D}$ *with* $r_1 > r_{sr} + D$ *denote the* D *re-appearances of the* $\texttt{ECS}(D+1)$*-common root* R *guaranteed by* $\Diamond\texttt{STABILITY}'$ *according to Definition 10. Then, no correct consensus algorithm under the message adversary* $\Diamond\texttt{STABLE}'(D)$ *can terminate strictly before round* r_D*.*

PROOF. We assume w.l.o.g. that $n > 4$ and $D < n - 3$. Furthermore, we do not let the adversary choose r_{sr} and $\{r_1, \ldots, r_D\}$, which results in an even stronger impossibility result.

First, let us define some communication graphs that we employ later on. For any graph $\mathcal{G}$, let $\widetilde{\mathcal{G}}$ denote the subgraph of $\mathcal{G}$ induced by $\Pi \setminus \{p_{n-1}, p_n\}$, augmented with the edge (p_{n-1}, p_n). Let $\overline{\mathcal{G}}$ be the same as $\widetilde{\mathcal{G}}$ except that the direction of this edge is reversed. In addition, let $\mathcal{G}'$ be a graph where $D + 2$ processes of $\Pi \setminus \{p_{n-1}, p_n\}$ constitute a chain C (actually, a tree), with head p_1 and two tails p_{n-3}, p_{n-2}, where the processes of $\Pi \setminus C$ only have incoming edges. Let $\mathcal{G}''$ be the same as $\mathcal{G}'$, except that the direction of all the edges in C is reversed and there is an edge $e = (p_{n-3}, p_{n-2})$ in $\mathcal{G}''$. Let $\mathcal{G}'''$ be the same as $\mathcal{G}''$ but with reversed direction of this edge e.

For a contradiction, assume that an algorithm $\mathcal{A}$ exists that solves consensus in a round $\tau \leqslant r_D - 1$. Then, $\mathcal{A}$ must solve consensus also in the following execution ε: Let

all processes start with input 0, and construct $\sigma = (\mathcal{G}^r)_{r=1}^{\infty}$ as follows: For $r \notin \{r_1, \ldots, r_D\}$ and $1 \leqslant r < r_{\mathrm{sr}}$ or $r > r_{\mathrm{sr}} + D$, if r is even, let $\mathcal{G}^r = \mathcal{G}''$; if r is odd, $\mathcal{G}^r = \mathcal{G}'''$. For $r_{\mathrm{sr}} \leqslant r \leqslant r_{\mathrm{sr}} + D$ or $r \in \{r_1, \ldots, r_D\}$, let $\mathcal{G}^r = \mathcal{G}'$. Clearly, $\sigma \in \Diamond\mathtt{STABLE}'(D)$. By validity and the assumptions on $\mathcal{A}$, all processes of Π must decide 0 by round τ.

We now define another execution ε', where all processes in $\Pi \setminus \{p_{n-1}, p_n\}$ start with 0 and p_{n-1} and p_n start with 1. The graph sequence σ' of ε' is the same as σ until round τ, except that every $\mathcal{G}^r$ of σ is replaced with $\widetilde{\mathcal{G}^r}$ if r is even and $\overline{\mathcal{G}^r}$ if r is odd. Moreover, $\mathcal{G}'$ in round $r_{\mathrm{sr}} + D$ is not only replaced with $\widetilde{\mathcal{G}'}$, but also augmented with a single edge (p_2, p_1). Finally, let the $\mathcal{G}^r$ of $(\mathcal{G}^r)_{r=\tau+1}^{\infty}$ in σ' be a star-graph with an out-edge from p_n to every process of Π. Again, note that $\sigma' \in \Diamond\mathtt{STABLE}'(D)$.

Observe that, in σ', for any round $r < r_D$, it holds that $p_1 \notin \mathrm{CP}_{p_{n-2}}^{r_{\mathrm{sr}}+D}(r)$. Hence, until round r, ε is indistinguishable for p_{n-2} from the execution ε'. In particular, p_{n-2} can not have learned about the existence of the edge (p_2, p_1) in $\mathcal{G}^{r_{\mathrm{sr}}+D}$. Therefore, since p_{n-2} decides 0 in round τ in ε, it does so also in ε'. This, however, means that p_n can never make a safe decision in ε': In order to satisfy agreement it should decide 0. However, since p_n never hears from process that had input 0, ε' is indistinguishable for p_n from an execution ε'', which has the same graph sequence σ' but where all processes have input 1. In order to satisfy validity, it should decide 1 in ε''. This provides the required contradiction. □

More efficient algorithms

Throughout our paper, we have assumed a full-information protocol where, every round, a process stores and forwards its entire known state history. While this is a convenient abstraction for introducing the fundamental concepts of our algorithm and a valid assumption for any impossibility result, it is of course highly unpractical.

We can name two major improvements related to this issue. For simplicity, we only discuss the graph approximation here and not the matrix $\mathtt{lock}_p$ of lock values. It is not hard to see that arguments for the former extend in a natural way to the latter.

First, it has already been shown, via the graph approximation algorithm used in [3], that it is sufficient to store and forward the local graph approximation history of each process in order to faithfully approximate the communication graph sequence. In round r, this requires up to $O(rn^2)$ local memory space at every process.

Second, the question arises whether it is indeed necessary to maintain (an approximation of) the entire communication graph sequence. In the case of $\Diamond\mathtt{STABLE}(D)$, it is perfectly possible to locally store and forward only a relatively small part of the graph approximation: Since the largest possible latency for a process to detect the start of a single-rooted graph sequence of duration $D+1$ is D rounds, it suffices to maintain only the last $2D+1$ rounds of the graph approximation history. This optimization yields a memory complexity of $O(Dn^2) = O(n^3)$ by Lemma 1.

In the case of $\Diamond\mathtt{STABLE}'(D)$, there is a tradeoff between the strength of the adversary and the memory complexity required by the algorithm. The principal issue is that if we allow the algorithm to purge the graph approximations for all but the last x rounds, then the adversary could generate a run with a "terminating" $\mathtt{ECS}(D+1)$-common root R with $r_D > r_{\mathrm{sr}} + D + x$, recall Definition 10. In this case, process $p \in \Pi$ in round r_D would not have its causal past down to round $r_{\mathrm{sr}} + D$ available, which is mandatory for detecting R.

A straightforward remedy would be an additional restriction to be enforced by the message adversary, which must ensure $r_D \leqslant r_{\mathrm{stab}} + D + x$ for some given additional parameter x. A message adversary weakened in such a way would entail a memory complexity of $O(xn^2)$ for our consensus algorithm.

7 Conclusion

We introduced an eventually stabilizing message adversary for consensus in a synchronous dynamic network with directed communication. Such a model closely captures the behaviour of a real network with arbitrarily irregular interconnection topology for a finite initial period, before it eventually starts to operate in a reasonably well-orchestrated manner.

Our message adversary eventually asserts a single strongly connected component without incoming edges from outside the component, which consists of the same set of processes, with possibly changing interconnection topology, either forever ($\Diamond\mathtt{STABLE}(D)$) or, in a generalized and stronger variant, for a certain number of (partly consecutive) rounds ($\Diamond\mathtt{STABLE}'(D)$). We established that no deterministic algorithm can terminate earlier than $2D+1$ rounds after stabilization in some execution under $\Diamond\mathtt{STABLE}(D)$, where D is the dynamic network diameter guaranteed by the message adversary, and provided a matching algorithm, along with its correctness proof, that even works under $\Diamond\mathtt{STABLE}'(D)$.

Part of our future work in this area will be devoted to finding even stronger message adversaries for stabilizing dynamic systems, and to the development of techniques for exploiting them algorithmically.

8 References

[1] Y. Afek and E. Gafni. Asynchrony from synchrony. In D. Frey, M. Raynal, S. Sarkar, R. Shyamasundar, and P. Sinha, editors, *Distributed Computing and Networking*, volume 7730 of *Lecture Notes in Computer Science*, pages 225–239. Springer Berlin Heidelberg, 2013.

[2] R. Baumann. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability*, 5(3):305–316, Sept. 2005.

[3] M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. In *Proceedings 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO'12)*, LNCS 7355, pages 73–84. Springer-Verlag, 2012.

[4] M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks, 2015. (to appear in Proc. NETYS'15, Springer LNCS; arxiv version: http://arxiv.org/abs/1501.02716).

[5] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *IJPEDS*, 27(5):387–408, 2012.

[6] É. Coulouma and E. Godard. A characterization of dynamic networks where consensus is solvable. In

Proceedings Structural Information and Communication Complexity - 20th International Colloquium (SIROCCO'13), Springer LNCS 8179, pages 24–35, 2013.

[7] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, Apr. 1988.

[8] C. Dyer and D. Rodgers. Effects on spacecraft & aircraft electronics. In *Proceedings ESA Workshop on Space Weather*, ESA WPP-155, pages 17–27, Nordwijk, The Netherlands, nov 1998. ESA.

[9] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.

[10] F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *STOC*, pages 513–522, 2010.

[11] F. Kuhn and R. Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, 2011.

[12] F. Kuhn, R. Oshman, and Y. Moses. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Prin ciples of distributed computing*, PODC '11. ACM, 2011.

[13] F. Kuhn, S. Schmid, and R. Wattenhofer. Towards worst-case churn resistant peer-to-peer systems. *Distributed Computing*, 22(4):249–267, 2010.

[14] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[15] F. Legendre, T. Hossmann, F. Sutton, and B. Plattner. 30 years of wireless ad hoc networking research: What about humanitarian and disaster relief solutions? What are we still missing? In *International Conference on Wireless Technologies for Humanitarian Relief (ACWR 11)*, Amrita, India, 2011. IEEE.

[16] C. Newport, D. Kotz, Y. Yuan, R. S. Gray, J. Liu, and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. *SIMULATION: Transactions of The Society for Modeling and Simulation International*, 83(9):643–661, Sept. 2007.

[17] M. Raynal and J. Stainer. Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors. In *Proceedings ACM Symposium on Principles of Distributed Computing (PODC'13)*, pages 166–175, 2013.

[18] N. Santoro and P. Widmayer. Time is not a healer. In *Proc. 6th Annual Symposium on Theor. Aspects of Computer Science (STACS'89)*, LNCS 349, pages 304–313, Paderborn, Germany, Feb. 1989. Springer-Verlag.

[19] U. Schmid, B. Weiss, and I. Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, 2009.

[20] M. Schwarz, K. Winkler, U. Schmid, M. Biely, and P. Robinson. Brief announcement: Gracefully degrading consensus and k-set agreement under dynamic link failures. In *Proceedings of the 33th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '14, pages 341–343, New York, NY, USA, 2014. ACM.